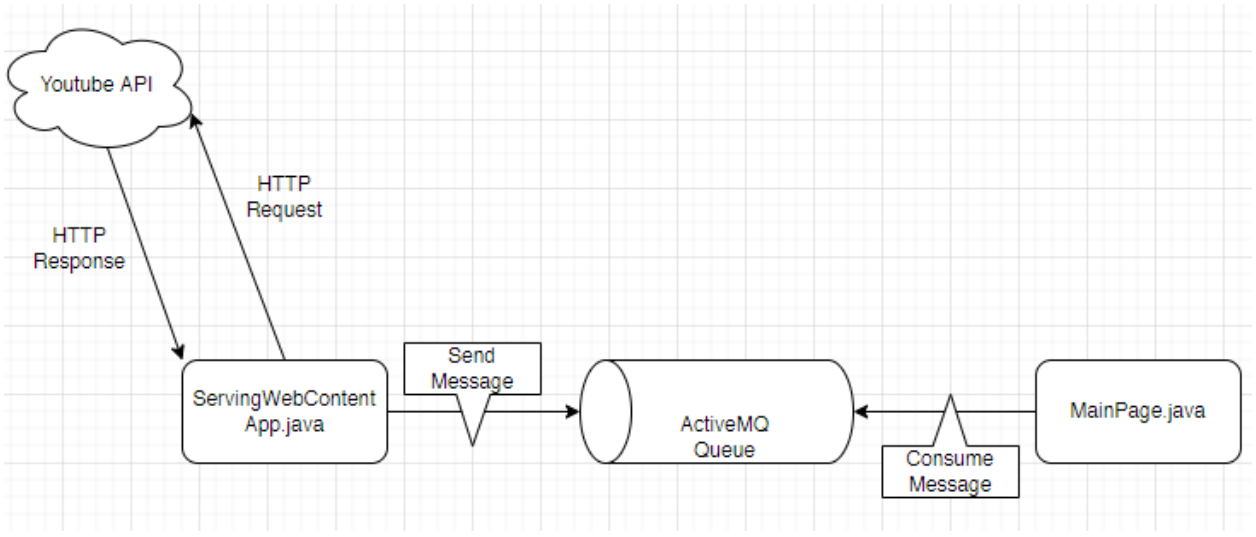
Design Patterns

1 design pattern was used during the implementation of the program.

Chain of Responsibility:

- The program utilizes multiple functions when executing the action of sending a JMS message to the queue. These functions are split into their own responsibilities which then, upon completion of their task will return data that the next function in the chain will use.
    - An example of which in the code base is for Part A. The API request is performed on the retrieveData function which will then serialize the data and return that data in an ArrayList of VideoData objects to the main “run” function
    - That data is then sent to the sendMessage function which will then send the message to the JMS queue

Schema



How to Compile

1. Download and unzip
2. Ensure the Java JDK version is at least Ver.11 and that the environment variable "JAVA_HOME" is set to the path to the Java JDK's folder.
3. Download ActiveMQ 5 and add path of the ActiveMQ 5's bin folder to the PATH environment variable
4. Launch ActiveMQ:
    a. Using command prompt use the command 'activemq start'
5. Launch the String Boot application:
    a. Using another command prompt navigate to the "demo" folder inside the main "Project YT" folder and use the command 'mvnw spring-boot:run'

How to Execute program

1. On a web browser go to "localhost:8080" to view the interface
2. (Optional)To view the JMS queues navigate to "localhost:8161/admin/queues/jsp":
    a. Login credentials
        i. Username: admin
        ii. Password: admin
3. Click on the "Run project and view logs" button to redirect to page that will run the program and display results

Reflection

Reflecting upon the task, a number of items I would consider doing differently:

Currently when passing the data to the JMS queue an XML string is passed before being parsed for data at the receiver. Next time I would like to spend the time to consider if there is an XML Object that can perform the same function to more closely match the requirement of "sending an XML message to the JMS Queue"

Furthermore, due to limited time constraints, I did not incorporate conventional design patterns during my implementation of the program. Next time I would definitely spend more time looking into which design patterns would be most effective to follow before the implementation stage has started.

A third change would be the page log used to keep track of, and print the actions taken, on the /mainPage. Currently the “A message has been send to queue: queueB” is a manual addition to the page log as the sendMessage function doesn’t return anything, and the ServingWebContentApplication class is a static class and so the log data won't be kept for that specific instance. This leads to no way of retrieving that page log data without more changes to the code base. Having to do this makes the process much more convoluted and I believe could have been achieved if proper design patterns were put in place during the implementation.